Innominate mGuard

Application Note: Rollout Support

mGuard smart
mGuard smart²

mGuard centerport

mGuard blade

mGuard industrial rs

mGuard pci

mGuard rs2000
mGuard rs4000

Innominate Security Technologies AG
Rudower Chaussee 13
12489 Berlin, Germany

Phone: +49 (0)30-921028 0
Fax: +49 (0)30-921028 020
contact@innominate.com
http://www.innominate.com

TABLE OF CONTENTS

1 Disclaimer 3

2 Introduction 4

3 Script *rollout.sh* 5

3.1 Bootstrap in detail 5

3.2 Remarks 6

3.3 Sample Script 1 (TFTP, dynamic) 7

3.4 Sample Script 2 (TFTP, static) 8

3.5 Sample Script 3 (WGET, dynamic) 9

4 mGuard rs2000/rs4000: Rollout using the SD Card 11

5 Further Readings 12

5.1 gaiconfig 12

1 Disclaimer

© Innominate Security Technologies AG

December 2011
“Innominate” and “mGuard” are registered trademarks of the Innominate Security Technologies AG. All other brand names or product names are trade names, service marks, trademarks, or registered trade marks of their respective owners.
mGuard technology is protected by the German patents #10138865 and #10305413. Further national and international patent applications are pending.
No part of this documentation may be reproduced or transmitted in any form, by any means without prior written permission of the publisher.
All information contained in this documentation is subject to change without previous notice. Innominate offers no warranty for these documents. This also applies without limitation for the implicit assurance of scalability and suitability for specific purposes.
In addition, Innominate is neither liable for errors in this documentation nor for damage, accidental or otherwise, caused in connection with delivery, output or use of these documents.
This documentation may not be photocopied, duplicated or translated into another language, either in part or in whole, without the previous written permission of Innominate Security Technologies AG.

2 Introduction

When rolling out projects with lots of devices it could be very time consuming to configure each mGuard manually through its web interface. This document describes how to pre-configure the mGuard devices before flashing them with the firmware.

The firmware can be downloaded at the Innominate website www.innominate.com, section *Downloads >> Firmware*. The firmware consists of the following files, depending on the mGuard product:

Product	Firmware files
mGuard smart **mGuard industrial rs** **mGuard PCI** **mGuard blade**	install.p7s jffs2.img.p7s
mGuard smart² **mGuard rs2000** **mGuard rs4000**	install-ubi.mpc83xx.p7s ubifs.img.mpc83xx.p7s
mGuard centerport	install.x86_64.p7s firmware.img.x86_64.p7s

Each mGuard product needs their corresponding firmware files for the flash procedure (refer to the list above). For a simplified notation, the following chapters mention only the firmware files *install.p7s* and *jffs2.img.p7s* representatively.

The flash procedure on the mGuard checks for the presence of the file *rollout.sh* in the root directory of the TFTP server. This file must be located in the same directory as the new firmware image files (*jffs2.img.p7s* and *install.p7s*). The file *rollout.sh* can be used for uploading either a configuration fragment to the devices (e.g. same firewall rules for all devices) or a complete configuration.

The following mechanisms are supported by the script *rollout.sh*:

1) **TFTP, dynamic**: A device specific configuration file *<serialnumber>.atv* is uploaded to the devices from the TFTP server's directory. Refer to chapter Sample Script 1 (TFTP, dynamic) for obtaining further information about how to configure the script *rollout.sh*.
2) **TFTP, static**: The same configuration is uploaded to all devices from the TFTP server's directory. This method is used for example to pre-configure all devices with the same firewall rules. Refer to chapter Sample Script 2 (TFTP, static) for obtaining further information about how to configure the script *rollout.sh*.
3) **WGET, dynamic:** A device specific configuration file is uploaded from an HTTP server on the TFTP server's host. During the flash procedure the mGuard transfers its serial number and additional information to the HTTP server. A script on the server is responsible to create the configuration file which is sent back to the mGuard. Refer to chapter Sample Script 3 (WGET, dynamic) for obtaining further information about how to configure the script *rollout.sh*.

3 Script *rollout.sh*

The mGuard provides basic means for supporting the rollout process, starting with firmware release 2.1.3. The implementation of the process itself depends on the user and is therefore not part of the standard distribution.

After flashing the firmware (please refer to the mGuard User Manual) the mGuard checks for the file *rollout.sh*. This file must be located in the same directory as the firmware image files (*jffs2.img.p7s* and *install.p7s*) on the TFTP server. If the file exists, it is uploaded to the mGuard and executed.

The file *rollout.sh* should be a UNIX shell script. The script can be used to retrieve the configuration data for the mGuard from the TFTP server and to start the mGuard configuration program (*gaiconfig*), which will configure the device accordingly. Chapter 3.1 contains more detailed information on the bootstrap process and on how to implement the script.

The contents of the script depend on the particular requirements of the user. Therefore the script is not provided by Innominate. The rollout support can be implemented in a way that all devices get the same configuration (refer to Sample Script 2 (TFTP, static)), or it can be implemented dynamically, configuring each device depending on its serial or flash number (refer to Sample Script 1 (TFTP, dynamic) and Sample Script 3 (WGET, dynamic)). Use these sample scripts as a reference for your own implementation.

3.1 Bootstrap in detail

1) Every bootstrapping file system image (*jffs2.img.p7s*) has a directory called */bootstrap/* that contains the bootstrapping control script (*/bootstrap/install.sh*). The control script deletes the entire */bootstrap* directory after the bootstrap process is successfully finished. Every time the kernel boots, it checks for an unfinished bootstrapping process by checking the existence of */bootstrap*.
2) The script *install.p7s* loads a script called *rollout.sh* from the TFTP server after loading and flashing the jffs2 file system image. *rollout.sh* is written to */bootstrap/rollout.sh* of the flash file system and executed from there. A missing *rollout.sh* script on the TFTP server is ignored.
3) The script *rollout.sh* has to create the script */bootstrap/preconfig.sh*. This script will be executed at the end of the bootstrap process.
4) The system starts bootstrapping after it has loaded and flashed *jffs2.img.p7s* completely, which is controlled by */bootstrap/install.sh*. After the installation has been successfully completed it checks for the existence of the configuration script */bootstrap/preconfig.sh*. If it does not exist, no further action is taken. If the script exists it will be executed. During the execution of *rollout.sh* the network is available, during the execution of *preconfig.sh* the mGuard is in standalone mode.

A successful execution of */bootstrap/preconfig.sh* is indicated by a specific flash pattern of the LED at the end of the flash procedure:

Product	Without pre-configuration	With pre-configuration
mGuard smart **mGuard smart²**	All three LED flash green simultaneously.	The outer of the three LED flash simultaneously in turn with the inner LED.
mGuard pci & **mGuard blade**	The mGuard restarts automatically.	The green LAN and WAN LED flash simultaneously in turn with the red WAN LED.
mGuard industrial rs	The modem, state and LAN LED flash green simultaneously.	The modem and LAN LED flash green simultaneously in turn with the status LED.
mGuard rs2000 **mGuard rs4000**	The LED *Stat*, *Mod* and *Sig* flash green simultaneously.	The LED *Stat* and *Sig* flash green simultaneously in turn with the *Mod* LED.

3.2 Remarks

gaiconfig can be called multiple times in *preconfig.sh*, first for a master configuration (e.g. *default.atv)* which contains a basic configuration to be used by all devices and second for the specific configuration *preconfig.atv*.

3.3 Sample Script 1 (TFTP, dynamic)

This is a sample script *rollout.sh* using tftp for downloading a device specific configuration file from the TFTP server. The script assumes that the configuration filename is <*serialnumber*>*.atv*. The script downloads the configuration file using tftp and creates the script *preconfig.sh* that calls the mGuard configuration program *gaiconfig*. *preconfig.sh* is automatically executed after all packets are installed successfully (refer to chapter 3.1).

```
#!/bin/sh -ex
# The IP address of the DHCP/TFTP server
# is supplied by install.p7s

server=$1
export PATH=/bin:/bootstrap
if test -x /bootstrap/sysmguard && \
test "`sysmguard kv`" -eq 6;
then
SERIAL=`sysmguard param oem_serial`
else
# /proc filesystem is needed
mount -t proc none /proc
SERIAL=`cat /proc/sys/mguard/parameter/oem_serial`
# /proc must be unmounted to enable unmounting of the jffs2 filesystem later.
umount /proc
fi

# This is the filename of the user supplied configuration file
# on the host in the TFTP-server directory
cfg_name="${SERIAL}.atv"

# fetch the configuration-file "preconfig.atv"
tftp -g -l - -r "$cfg_name" "${server}" | dd bs=1M of=/bootstrap/preconfig.atv

# create a small configuration-script that installs the
# configuration fetched from ${server}
cat >/bootstrap/preconfig.sh <<EOF
#!/bin/sh
modprobe param_dev 2>/dev/null
gaiconfig --silent --set-all < /bootstrap/preconfig.atv
EOF

# Make it executable. It will be executed after all packets
# are installed completely.
chmod 755 /bootstrap/preconfig.sh
```

The script *rollout.sh* must be stored in UNIX format.

3.4 Sample Script 2 (TFTP, static)

This is a sample script *rollout.sh* using tftp for downloading a standard configuration file for all devices from the TFTP server. The script assumes that the configuration filename is *preconfig.atv*. The script downloads the configuration file using tftp and creates the script *preconfig.sh* that calls the mGuard configuration program *gaiconfig*. *preconfig.sh* is automatically executed after all packets are installed successfully (refer to see chapter 3.1).

```
#!/bin/sh -ex
# The IP address of the DHCP/TFTP server
# is supplied by install.p7s
server=$1

# This is the filename of the user supplied static configuration file
# on the host in the TFTP-server directory
cfg_name=preconfig.atv
export PATH=/bin:/bootstrap

# fetch the static configuration-file "preconfig.atv"
tftp -g -l - -r "$cfg_name" "${server}" | dd bs=1M of=/bootstrap/preconfig.atv

# create a small configuration-script that installs the
# configuration fetched from ${server}
cat >/bootstrap/preconfig.sh <<EOF
#!/bin/sh
modprobe param_dev 2>/dev/null
gaiconfig --silent --set-all < /bootstrap/preconfig.atv
EOF

# Make it executable. It will be executed after all packets
# are installed completely.
chmod 755 /bootstrap/preconfig.sh
```

The script *rollout.sh* must be stored in UNIX format.

3.5 Sample Script 3 (WGET, dynamic)

This is a sample script *rollout.sh* using wget for downloading a device specific configuration file from a HTTP server on the TFTP server. The script assumes that a script on the HTTP server (*preconfig.cgi* in this example) identifies the device using the serial number. The mGuard transmits its serial number and additional information to the HTTP server. The script on the server side (*preconfig.cgi* in this example) is responsible for creating the script *preconfig.sh* which is sent to the mGuard and installed in the mGuard's */bootstrap* directory. *preconfig.sh* calls the mGuard configuration program *gaiconfig* and is automatically executed after all packets are installed successfully (refer to chapter 3.1).

```
#!/bin/sh -ex
# The IP address of the DHCP/TFTP server
# is supplied by install.p7s
server=$1
export PATH=/bin:/bootstrap

if test -x /bootstrap/sysmguard && \
test "`sysmguard kv`" -eq 6;
then
FID=`sysmguard flash factory`
HW=`sysmguard hw revision`
SERIAL=`sysmguard param oem_serial |sed 's/ /\%20/g'`
CPU=`sysmguard hw cpu |sed 's/ /\%20/g'`
NAME=`sysmguard param name |sed 's/ /\%20/g'`
else
# /proc filesystem is needed
mount -t proc none /proc
# fetch information about the system and replace spaces by %20
FID=`cat /proc/sys/mguard/hw/factoryregister`
HW=`cat /proc/sys/mguard/hw/revision`
SERIAL=`cat /proc/sys/mguard/parameter/oem_serial |sed 's/ /\%20/g'`
CPU=`cat /proc/sys/mguard/hw/processortype |sed 's/ /\%20/g'`
NAME=`cat /proc/sys/mguard/parameter/name |sed 's/ /\%20/g'`
# /proc must be unmounted to enable unmounting of the jffs2 filesystem later.
umount /proc
fi

SW=`cat /etc/version`

# call the configuration CGI with all the information about us and
# store the result as /bootstrap/preconfig.sh
# The configuration in this example is done by a http-server on
# the DHCP/TFTP server host.
wget -O /bootstrap/preconfig.sh "http://$server/cgi-
bin/preconfig.cgi?FID=$FID&HW=$HW&SW=$SW&CPU=$CPU&NAME=$NAME&SERIAL=$SERIAL"

# Make it executable. It will be executed after all packets
# are installed completely.
chmod 755 /bootstrap/preconfig.sh
```

The script *rollout.sh* must be stored in UNIX format.

The corresponding CGI script has to retrieve the configuration from a database (or from configuration files) and should create a shell script *preconfig.sh* that calls the mGuard configuration program *gaiconfig*:

```
#!/bin/sh
modprobe param_dev 2>/dev/null
gaiconfig --silent --set-all <<EOF
NETWORK_HOSTNAME=mGuard0232
MY_LOCAL_IP=192.168.30.40
...
EOF
```

In this example the configuration values *192.168.30.40* and *mGuard0232* have to be retrieved from a database (or configuration files) on the HTTP server, depending on the information received from the mGuard.

4 mGuard rs2000/rs4000: Rollout using the SD Card

The rollout of the mGuard rs2000/rs4000 can be performed using the SD card. The SD card must contain the subdirectories ***Firmware*** and ***Rescue Config***.

The subdirectory ***Firmware*** must contain the firmware image files *install-ubi.mpc83xx.p7s* and *ubifs.img.mpc83xx.p7s*:

/Firmware/install-ubi.mpc83xx.p7s
/Firmware/ubifs.img.mpc83xx.p7s

The subdirectory ***Rescue Config*** must contain the files *preconfig.atv* and *preconfig.sh*:

/Rescue Config/preconfig.atv
/Rescue Config/preconfig.sh

The file *preconfig.atv* is the configuration profile which should be applied to the mGuard. The file *preconfig.sh* must contain the following two lines:

```
#!/bin/sh
exec gaiconfig --silent --set-all < /bootstrap/preconfig.atv
```

The file *preconfig.sh* must be stored in UNIX format.

After preparing the SD card accordingly and inserting it into the SD slot of the mGuard rs2000/rs4000, press the rescue button and keep it pressed until the LED *STAT*, *MOD* and *SIG* shine continuously green.
Now the flash procedure starts. The mGuard is flashed with the provided firmware and the configuration profile is applied.
The procedure is finished when the LED *STAT* and *SIG* flash green simultaneously in turn with the *MOD* LED.
Please refer to the *mGuard User Manual* to obtain further information about the flash procedure.

5 Further Readings

For more detailed information please refer to the *mGuard User Manual*.

5.1 gaiconfig

The *Generic Administration Interface's* (GAI) purpose is to provide an interface to configure the mGuard from the command line. *gaiconfig* is the command line tool to get and set variables in all configuration files managed by GAI. Services are stopped and started as defined in the registry before the program exits. Upon successful completion a value of 0 is returned. This command can be used by all users who are members of the group *admin*. Here is a selection of *gaiconfig* commands:

--help	Print a short description of the command.
--get-all	Dump all read-write variables on stdout, using the Attribute-Type-Value (ATV) object serialization language.
--set-all	Set all write-only and read-write variables mentioned in the file in Attribute-Type-Value (ATV) object serialization language from stdin.
--silent	Just rewrite the configuration files but don't stop and start depending services.

For further information on *gaiconfig* please refer to the *gaiconfig User Guide*.